Massachusetts Institute of Technology
Charles Stark Draper Laboratory
Cambridge, Massachusetts

LUMINARY Memo # 143

TO: Distribution
FROM: Allan Klumpp
DATE: March 24, 1970
SUBJECT: Automatic P66

HISTORICAL PREFACE

Program P65 was designed to guide the lunar module during the last 100 feet or so of a lunar landing. It was designed to produce a vertical descent by nulling the horizontal components of velocity and providing a constant negative vertical component. Program P66 was designed as an alternative to program P65 to be used for the last several hundred feet of a lunar landing. It was designed to produce a vertical component of velocity controlled by astronaut commands via a Rate Of Descent switch, and Attitude Hold / Attitude Rate Command capability which would permit the astronaut to control the attitude such as to null the horizontal components of velocity manually. Program P66 was used on Apollos 11 and 12. P65 was available but never used.

The P65 guidance equation, which is finally being eliminated, is the precursor of the new horizontal algorithm of the auto P66 mode. Some background on the P65 algorithm should be interesting and provide notivation for the P66 horizontal algorithm.

My first knowledge of the P65 guidance equation was in August 1965 when I produced it for the sole purpose of making simulations work. It was the simplest conceivable stopgap measure to allow deferment of the minor problem of vertical descent until some of the major problems were solved. I never would have believed at the time that this inadequate algorithm would survive to be coded in the guidance computer of the first two lunar landing missions. But there was never much incentive to improve the algorithm because it was the expressed intent of every astronaut queried not to use P65 whether it was improved or not.

The P65 equation produced a vector acceleration command proportional to the vector velocity error, and therefore, to a linear continuous system approximation provided an exponential decay of the velocity error. It reads

$$\underline{A}_c = -(\underline{v} - \underline{v}_c)/\tau,$$

where $\underline{A}_c$ is the commanded acceleration vector, $\underline{v}$ is the current velocity vector, $\underline{v}_c$ is the commanded velocity vector, and τ is the time constant. Assuming $\underline{v}_c$ is constant and the commanded acceleration is achieved continuously, the solution for the velocity error is:

$$\underline{v}_e = \underline{v} - \underline{v}_c = \underline{v}_{eo}\, e^{-t/\tau},$$

where $\underline{v}_{eo}$ is the velocity error at time t = 0.

This P65 equation is inadequate because the two second sample interval combined with the slow attitude response about the two horizontal axes forbids a reasonable compromise between speed of response and well damped behavior. For well damped behavior, τ must be several times the combined sample interval and system delays; the result is too slow. In missions τ has been set at 8 or 10 seconds.

For a feedback system of a given configuration, there is frequently a compromise between damping and speed. One can be traded for the other similarly to the tradeoff between gain and bandwidth in amplifier design. In the case of the horizontal channels of the vertical descent guidance algorithm, there is not much we can do to reduce the system delays, but if we could change the configuration of the guidance algorithm such as to produce inherently better damping, we could then trade a portion of the damping improvement for faster response. For a position control system, velocity feedback provides damping. For a velocity control system, acceleration feedback provides damping. We have a velocity control system.

In February, 1969 I suggested to my colleague Nicholas Pippenger that he investigate acceleration feedback for the P65 equation. I envisioned feeding back into the horizontal channels an acceleration derived from spacecraft attitude, thrust, and mass. Starting with some ideas proposed by Jerrold Suddath of MSC (Ref. 1), Nick produced an algorithm at least as effective as my proposal and far simpler to implement than either suddath's or mine. Nick proposed to feed back the prior acceleration command. Because of the system delays, the prior command is a very good measure of the current acceleration. In fact Nick has studied a class of algorithms which save and feedback the acceleration commands from several past cycles.

After reading the accounts from Apollos 11 and 12 of the difficulty of determining translational rates during the final 100 feet or so of vertical descent, it seemed to me that we could provide future missions with an automatic mode for nulling horizontal velocity in P66 based on the knowledge of velocity by the primary system which is undegraded by the dust. My colleague Donald Eyles had produced the germ of the idea when he suggested the capability to return to P65 from P66, but that proposal would produce poor horizontal response and forfeit rate of descent control. An automatic P66 mode could incorporate the algorithm conceived by Nick Pippenger and continue to provide rate of descent control. The question of whether such a mode would be desirable was put to the Apollo 12 crew in the crew de-briefings on 9 December, 1969. On December 12, MIT was asked by NASA whether we could provide the new capability in time for Apollo 13. NASA also suggested a significant improvement, i.e. to provide attitude error needles in P66 attitude hold as well as in P66 auto. The primary doubts we had were whether there was computation time sufficient to compute the horizontal commands and whether the slower horizontal channels could be satisfactorily married to with the faster vertical channel. By December 15 we felt we had several workable solutions to both problems, and since these solutions consisted largely of re-connecting existing programs, we ventured we could produce a sample Luminary program for testing by NASA in short order. We delivered the test program on December 22. The guidance algorithm of this first version has not been changed in any way.

FEATURES OF THE NEW P66 PROGRAM

1. Attitude Hold Mode

The rate of descent (ROD) algorithm of the prior P66 program has been retained without change for the vertical channel of P66 Attitude Hold and P66 Auto. Thus P66 Attitude Hold provides all of the features of the prior P66 program. In addition it provides attitude error needles which work identically to the needles in other powered flight programs, e.g. P64. They display the error between the current attitude and the attitude required to null the horizontal velocity (the attitude which would be commanded by P66 auto). P66 Attitude Hold may be entered in any of three ways; 1) by switching to Attitude Hold in program P64 and waiting until the automatic program change to P66; 2) by switching to Attitude Hold in P63 or P64 and manipulating the rate of descent (ROD) switch; 3) by switching to Attitude Hold while in P66 Auto.

2. Auto Mode

This mode provides fast, well damped, attitude limited nulling of the horizontal velocity error and control over the descent rate via the ROD switch. The horizontal dynamics are controlled by two erasable constants TAUHZ and QHZ, and the attitude limit is controlled by the erasable AHZLIM. In addition P66 auto provides the normal powered flight attitude error needles. In future revisions P66 Auto may provide the capability to increment the horizontal command via the hand controller in the same way redesignations are made in P64. This will permit the astronaut to remove any PGNCS velocity bias he observes. The horizontal velocity increment will be set by an erasable. P66 Auto may be entered in either of two ways; 1) by waiting for the automatic program change at the conclusion of P64 while in Auto; 2) by switching to Auto while in P66 Attitude Hold. Thus, as in P64, there is a two way path between Auto and Attitude Hold which may be traversed as often as one wishes.

3. New Priority Structure

Testing of an early revision of the new program revealed violent throttle behavior under time loss stress testing (TLOSS) which simulates unexpectedly heavy counter activity or computation demand. The difficulty, described in Ref. 2, is caused by the failure of one servicer cycle to finish

before the subsequent servicer cycle is begun. Thus, two servicer jobs could iterfere with one another's erasables in the ROD equations. To avoid such conflicts:

A. The priority of the servicer job is raised from 20 to 21 at the start of each P66 pass, which assures one servicer job will finish before the subsequent servicer is permitted to start.

B. The RODTASK starts the ROD job at priority 22. This permits the ROD job to interrupt a servicer job anywhere (except in the ROD equations because of item C).

C. At the start of the ROD equations each servicer job is raised from priority 21 to 23, and each ROD job is raised from 22 to 23. This assures that only one job at a time can pass thru the ROD equations.

4. P66 Omissions and Padload 2LATE466

Item 3 above delays the start of a servicer cycle whenever the previous cycle is unfinished. If servicer persistently runs slower than expected, it will start increasingly late. Because the accelerometers are always read on time, we must prevent servicer starting so late as to occasionally miss using one pass of accelerometer data or the associated time tag called PIPTIME1. Thus when servicer gets behind to the point that on the subsequent pass the accelerometers would be in danger of being read before COPY-CYCL, P66 is omitted to permit the subsequent servicer to start on time. The criterion is the erasable 2LATE466, "too late for 66" which is the time relative to PIPTIME beyond which P66 is omitted. The test is made at the start of the P66 guidance equations. Thus with 2LATE466 loaded as 1.5 seconds, if the servicer program exits to P66 later than 1.5 seconds past PIPTIME, P66 will omit the horizontal and vertical equations and exit directly to the displays.

5. Alarm 01466 and Padload TOOFEW

In case of extremely severe TLOSS or computation demand, item 4 above can cause P66 to be omitted so frequently as to produce throttle or attitude control instability, or it can even lock out P66 altogether. Thus, when stability is endangered by too frequent omissions, alarm 01466 is issued, and P66 is omitted anyway. The criterion for issuing the alarm is the

erasable TOOFEW, which should be loaded as one less than the minimum number of successful throttlings between a P66 omission. Note that each time P66 is omitted once, the throttle is omitted twice, because when P66 is omitted it does not set up the subsequent RODTASK. TOOFEW can be thought of as "two few throttlings between a pair of throttle omissions". Thus, with TOOFEW loaded as three, four successful throttlings (two successful P66 passes) are required between omitting a pair of throttlings (omitting one P66 pass). Omitting more frequently produces alarm 01466.

6. X Axis Override

X Axis Override is permitted in both P66 Auto and Attitude Hold.

7. Throttle Modes

P66 now permits one to switch back and forth between manual and automatic throttle as often as one wishes. In revision 131 of Luminary, if the astronaut should bring the spacecraft to a desired descent rate using the manual throttle and then switch back to automatic throttle in order to regain ROD switch control, the commanded descent rate which existed prior to switching to manual throttle would still exist. This command would cause the throttle to drop and the spacecraft to revert to the prior descent rate, which could be dangerous at low altitude. The new program checks the automatic throttle discrete and resets the rate of descent command (VDGVERT) equal to the current rate of descent (HDOTDISP) whenever the throttle is in manual.

8. Landing Radar Off

The radar is turned off at an altitude controlled by the erasable constant HLROFF, fulfilling PCR 942.

9. Response to Touchdown

P66 Auto maintains the attitude error needles and continues issuing attitude commands, and P66 Attitude hold maintains the attitude error needles, until the astronaut indicates touchdown by both; 1) "Proceeding" on the flashing V06 N60 display and 2) switching the engine arm switch to off. At this point P66 stops the above functions and causes the DAP to make one pass thru DAPIDLER every two seconds. This may allow sporadic jet firing for two seconds, but will stop the jets completely after two seconds.

10. New Overflow Subroutine

A new subroutine has been coded for responding to overflows which occur anywhere in the descent, including P63, P64, or P66. This subroutine issues alarm 01410, does a STOPRATE in INHINT, and resets the overflow indicator. The INHINT is desirable to prevent the autopilot from interrupting the STOPRATE. Previously the INHINT was not done and the overflow indicator was not reset.

11. INHINT in RATESTOP

The descent subroutine RATESTOP, which is used whenever STEERSW is off indicating insufficient thrust, or following the 01406 alarm indicating the time-to-go equations fail to converge, now does the STOPRATE under INHINT.

12. Order of Overflow and STEERSW Checks, Deletion of Automatic Attitude Check before doing STOPRATE

Overflow is now tested before STEERSW, reversing the previous order, so that overflow will be checked even if STEERSW is off. In addition, the STOPRATE in case there is overflow or in case STEERSW is down is no longer omitted in Attitude Hold. FINDCDUW is supposed to do a STOPRATE in Attitude Hold, but since FINDCDUW is also omitted in the overflow case, the STOPRATE might never have been done in Luminary 131. In the case when STEERSW is down, the STOPRATE is superfluous since one is done by Servicer.

13. Deletion of INHINT and ZATTEROR at STOPFIRE

The ZATTEROR, which was previously done in response to the proceed at touchdown, does not succeed in stopping hard-on attitude jets, whereas the measures described in item 9 above do stop jet activity. Therefore the ZATTEROR and the associated INHINT have been deleted.

14. Deletion of P65

The coding associated with P65 has been deleted where necessary to provide room for the Auto P66 coding. The remaining P65 coding will be deleted from future revisions of Luminary.

P66 HORIZONTAL GUIDANCE ALGORITHM

The simplest type of algorithm, such as was used in P65, produces a vector acceleration command proportional to the vector velocity error, with identical gains in all components. Nick Pippenger proposed originally (for a modified P65 type program) to supplement that result with a negative fraction of the previous vector acceleration command, retaining equal gains in all components. For a P66 type program with ROD inputs, retaining equal gains for the three components is neither feasible nor necessary. It is not feasible because there is insufficient computation time to compute the horizontal commands as frequently as the vertical commands are computed, and the attitude changes which modulate the acceleration for horizontal control cannot be made anywhere near as fast as the throttle changes which modulate the acceleration for vertical control. Equal gains are not necessary because we do not require as speedy response to small horizontal velocity errors as we require to large ROD inputs.

These considerations dictate an algorithm in which the horizontal commands are computed once per two seconds and vertical commands once per second. The vertical channel can hopefully be left entirely unchanged.

One important design question was whether the attitude commands (horizontal channels) should be generated preceding or following the throttle commands (vertical channel). We decided to process the horizontal channels before the vertical channel for the following reasons:

1. The existing P66 vertical algorithm solves for the state inputs at the time it is processed and is therefore insensitive to the additional time delay in processing the horizontal channels first.

2. Assuming the horizontal channels use the velocity data generated by the normal state vector update routine, processing the horizontal channels first avoids the computation delay (about 300 milliseconds) of the vertical channel, thus improving the horizontal dynamics. In the future we could consider using for the horizontal channels the velocity vector generated by the vertical channel, in spite of item 3 below.

3. Because available computation time requires the horizontal channel to be updated only half as often as the vertical, the simplest possible organization is to locate the horizontal coding at a point which is accessed only once per two seconds. Thus, odd and even cycle tests are avoided.

Another important design question was whether we should use the vertical component of thrust acceleration commanded as one of the inputs for attitude control (as per P65) or whether we should assume the vertical thrust acceleration identical to lunar gravity, ignoring the thrust modulation required to alter the descent rate. Ignoring this vertical thrust modulation will modulate the gains in the horizontal channels and we will achieve the commanded horizontal acceleration only when descending at a constant rate.

In the case when there is a substantial upward acceleration, e.g. when the throttle is above about 60%, the horizontal channels become unstable and the attitude oscillates divergently. For example, at the soft throttle stop the attitude oscillates with a period of about 14 seconds and a divergence of about 20% per cycle. At maximum thrust the period is about 12 seconds and the divergence is perhaps 30% per cycle. The erasable padload AHZLIM limits the growth of such attitude oscillations in the same way it limits the deviation from an erect attitude in the normal case when there is a substantial velocity error, except for some attitude overshoot due to control limitations of theDAP. These attitude oscillations are likely to occur only if one should choose to abort in Auto P66 from a point where sufficient propellent remained to permit such oscillations to develop. In this unlikely case, one would leave a very crooked con-trail.

In spite of the above considerations, all of which were anticipated, we decided to ignore thrust modulation in the horizontal channels because:

1. To use the vertical thrust acceleration commanded as the vertical input for the thrust direction command (as per P65) would produce attitude modulation in response to astronaut ROD commands. This would be disconcerting and it would increase consumption of attitude propellant.

2. The vertical channel can modulate the thrust so much faster than the horizontal channels can modulate the attitude that when the thrust is being modulated we could not react fast enough to achieve the commanded horizontal acceleration if we tried.

3. The vertical thrust acceleration must be nearly constant because any significant variation produces a large change in the descent rate in a very short time (e.g. a 20% variation produces over 1 ft/sec^2 vertical acceleration).

4. The small and short duration variations of the gains in the horizontal channels due to modulation in the vertical channel do not significantly alter the dynamic response of the horizontal channels.

5. To use the vertical command, we would either have to use the command saved from the previous cycle or generate the horizontal commands after the vertical which we had decided not to do.

The following table lists some gain pairs supplied by Nick Pippenger which produce increasingly faster and less well damped horizontal response. The first row produces an effective time constant of about 4 seconds and essentially dead beat response. It has been used in all simulations to date.

TAUHZ (sec)	QHZ	TAUHZ (oct)	QHZ (oct)
5	.4	07640	14632
4	.45	06200	16315
3.448	.50	05306	20000
2.857	.525	04356	20632

The following table lists the erasable loads corresponding to various limits of pitch and roll angles. The 20^o limit has been used in most simulations to date.

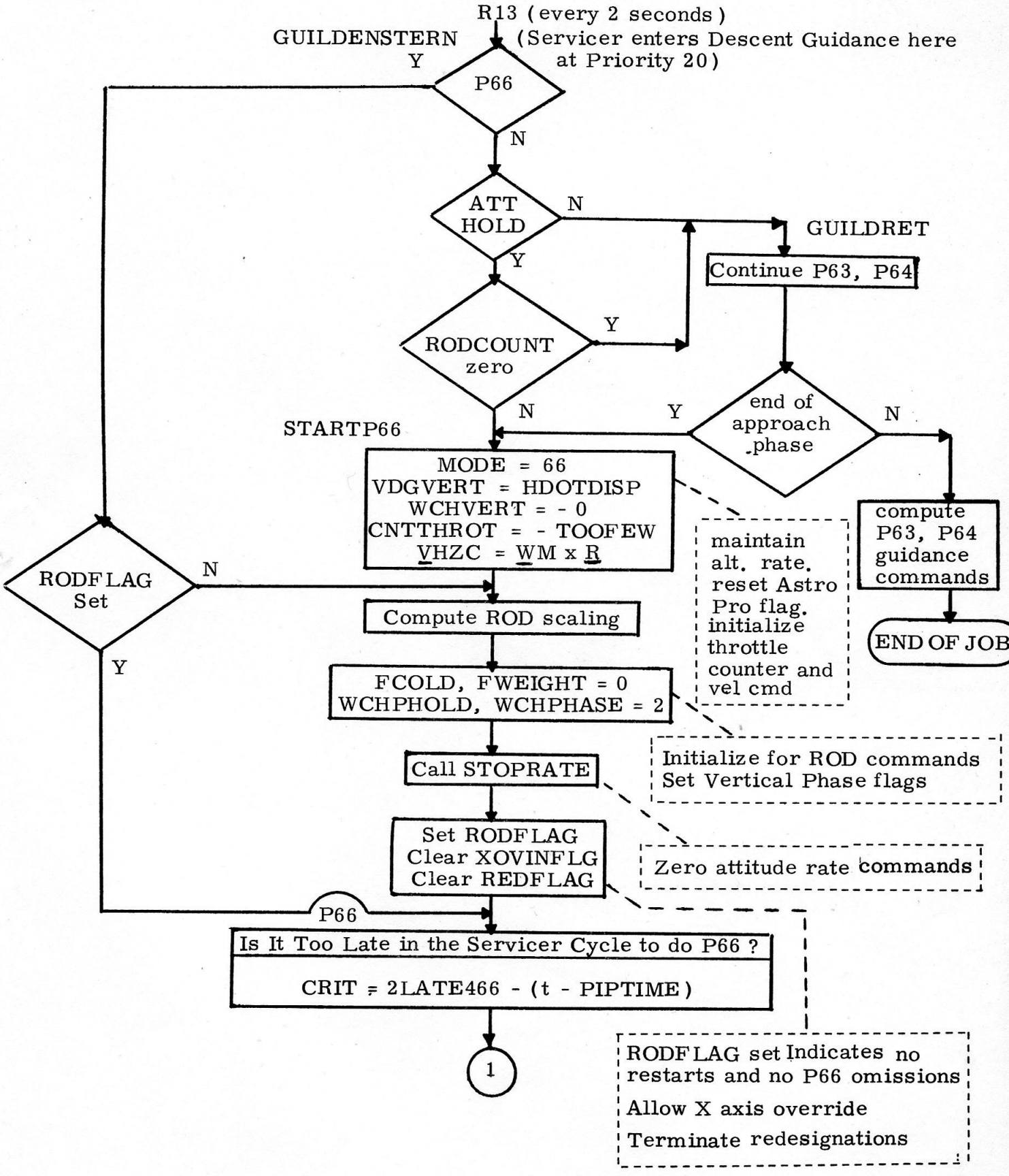
Angular Limit	AHZLIM (M/sec^2)	AHZLIM (oct)
15^o	.435	00013
20^o	.591	00017
25^o	.757	00024
30^o	.936	00031
35^o	1.135	00036
40^o	1.360	00044
45^o	1.623	00053
60^o	2.810	00112

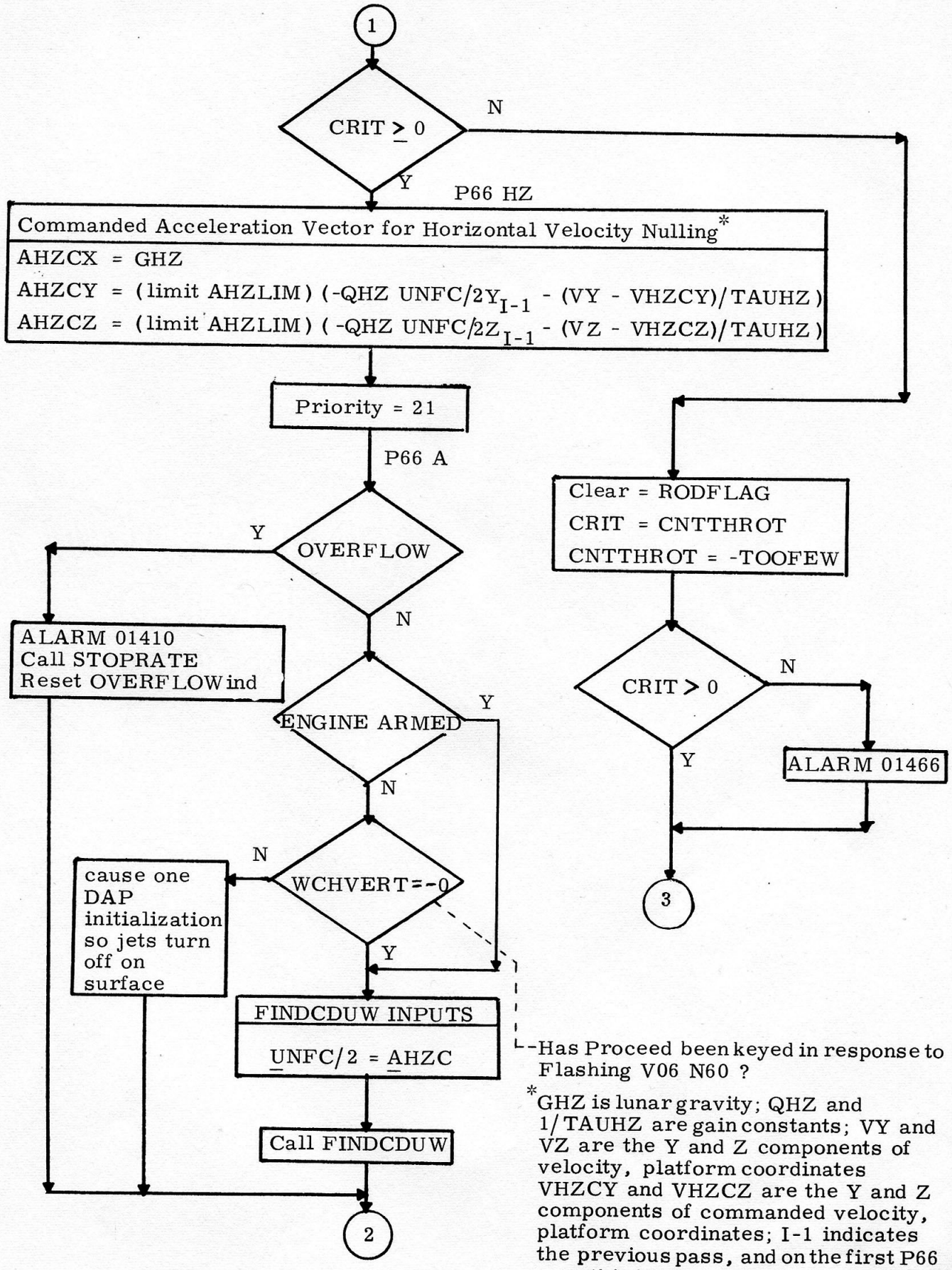
The following flow chart defines the Auto P66 program as presently coded for Apollo 13. The notation is identical to the LGC program.

References:

1. Suddath, J.H., "Guidance and Control Systems Interaction in P64 and P65", NASA Memo EG23-69-39, February 10, 1969.

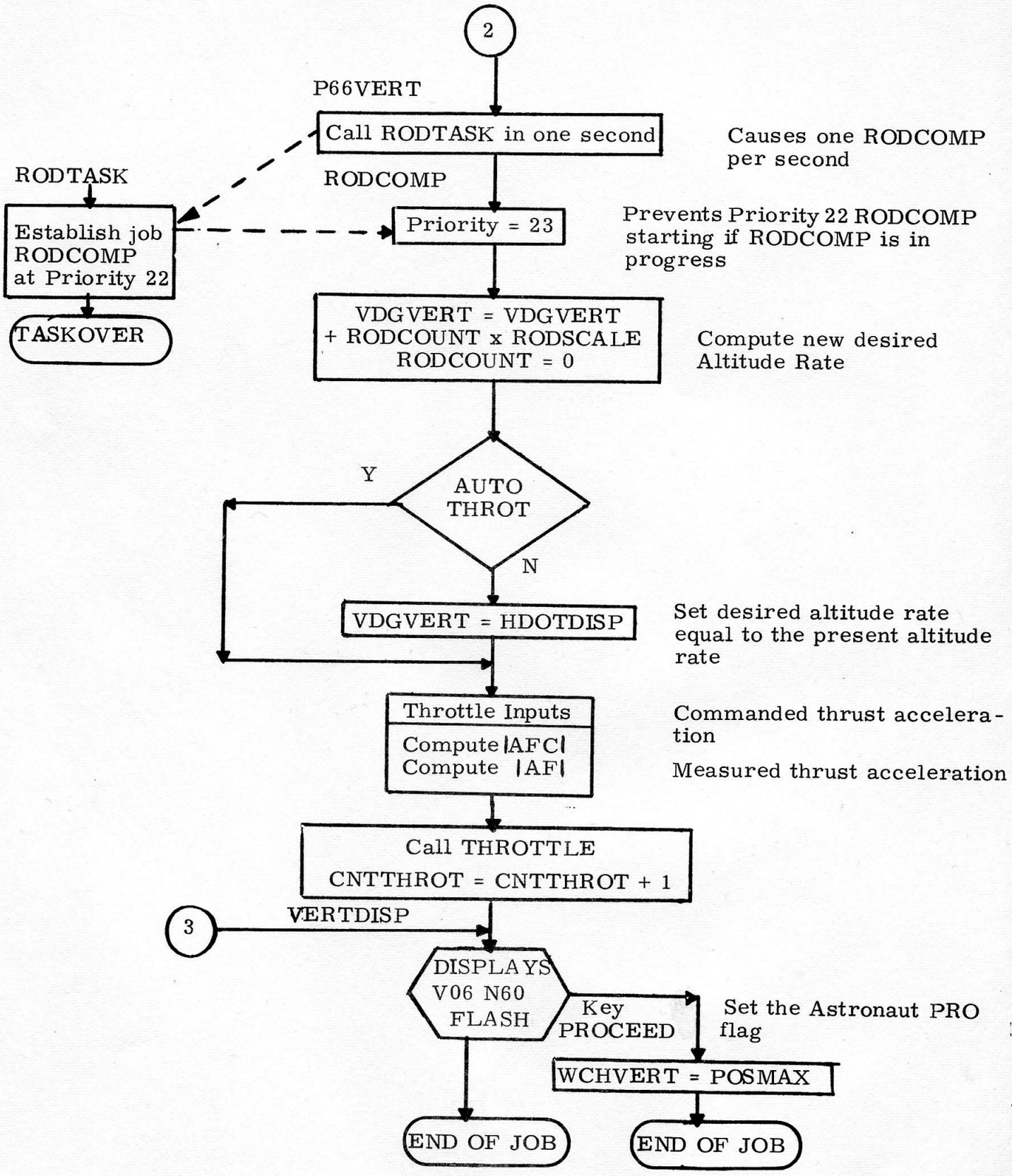
2. Klumpp, A., "A Collection of the Known Manifestations of Time Loss in Luminary Revision 131 and LM131 revision 001 - Suggested Work-Around Procedures", Luminary Memo #140, March, 1970.


R13 (every 2 seconds)
GUILDENSTERN
(Servicer enters Descent Guidance here at Priority 20)
Y
P66
N
ATT HOLD
N
GUILDRET
Y
Continue P63, P64
RODCOUNT zero
Y
N
Y
end of approach phase
N
STARTP66
MODE = 66
VDGVERT = HDOTDISP
WCHVERT = - 0
CNTTHROT = - TOOFEW
VHZC = WM x R
maintain alt. rate. reset Astro Pro flag. initialize throttle counter and vel cmd
compute P63, P64 guidance commands
END OF JOB
RODFLAG Set
N
Compute ROD scaling
Y
FCOLD, FWEIGHT = 0
WCHPHOLD, WCHPHASE = 2
Initialize for ROD commands
Set Vertical Phase flags
Call STOPRATE
Set RODFLAG
Clear XOVINFLG
Clear REDFLAG
Zero attitude rate commands
P66
Is It Too Late in the Servicer Cycle to do P66 ?
CRIT = 2LATE466 - (t - PIPTIME)
1
RODFLAG set Indicates no restarts and no P66 omissions
Allow X axis override
Terminate redesignations



Has Proceed been keyed in response to Flashing V06 N60 ?

*GHZ is lunar gravity; QHZ and 1/TAUHZ are gain constants; VY and VZ are the Y and Z components of velocity, platform coordinates VHZCY and VHZCZ are the Y and Z components of commanded velocity, platform coordinates; I-1 indicates the previous pass, and on the first P66 pass this indicates the final P64 pass. (LIMIT AHZLIM) indicates the content of the subsequent parentheses is magnitude limited to AHZLIM.


2
P66VERT
Call RODTASK in one second
Causes one RODCOMP per second
RODTASK
RODCOMP
Establish job RODCOMP at Priority 22
Priority = 23
Prevents Priority 22 RODCOMP starting if RODCOMP is in progress
TASKOVER
VDGVERT = VDGVERT + RODCOUNT x RODSCALE
RODCOUNT = 0
Compute new desired Altitude Rate
Y
AUTO THROT
N
VDGVERT = HDOTDISP
Set desired altitude rate equal to the present altitude rate
Throttle Inputs
Compute |AFC|
Compute |AF|
Commanded thrust acceleration
Measured thrust acceleration
Call THROTTLE
CNTTHROT = CNTTHROT + 1
3
VERTDISP
DISPLAYS V06 N60 FLASH
Key PROCEED
Set the Astronaut PRO flag
WCHVERT = POSMAX
END OF JOB
END OF JOB